remote sensing

MDPI

Article

PulseNetOne: Fast Unsupervised Pruning of Convolutional Neural Networks for Remote Sensing

David Browne [1,*], Michael Giering [2] and Steven Prestwich [1]

[1] Insight Centre for Data Analytics, University College Cork, Cork City T12 XF62, Ireland; steven.prestwich@insight-centre.org

[2] United Technologies Research Centre, Penrose Wharf Business Centre, Penrose Quay, Cork City T23 XN53, Ireland; gierinmj@utrc.utc.com

* Correspondence: david.browne@insight-centre.org

Received: 21 February 2020; Accepted: 24 March 2020; Published: 29 March 2020

Abstract: Scene classification is an important aspect of image/video understanding and segmentation. However, remote-sensing scene classification is a challenging image recognition task, partly due to the limited training data, which causes deep-learning Convolutional Neural Networks (CNNs) to overfit. Another difficulty is that images often have very different scales and orientation (viewing angle). Yet another is that the resulting networks may be very large, again making them prone to overfitting and unsuitable for deployment on memory- and energy-limited devices. We propose an efficient deep-learning approach to tackle these problems. We use transfer learning to compensate for the lack of data, and data augmentation to tackle varying scale and orientation. To reduce network size, we use a novel unsupervised learning approach based on k-means clustering, applied to all parts of the network: most network reduction methods use computationally expensive supervised learning methods, and apply only to the convolutional or fully connected layers, but not both. In experiments, we set new standards in classification accuracy on four remote-sensing and two scene-recognition image datasets.

Keywords: pruning networks; network compression; remote-sensing image classification; transfer learning; Convolutional Neural Network; pre-trained AlexNet; pre-trained VGG16

1. Introduction

Remote-sensing image classification has gained in popularity as a research area, due to the increase in availability of satellite imagery and advancements in deep-learning methods for image classification. A typical image contains objects and natural scenery, and the algorithms must understand which parts of the image are important and relate to the class, and which parts are unrelated to the class. Scene classification has been applied to various industrial products such as drones and autonomous robots, to improve their predictability at understanding scenes.

Much early work concentrated on using hand-crafted methods such as color histograms, texture features, scale-invariant feature transform (SIFT) or histograms of oriented gradient (HOG). Color histograms were very simple to implement but, though translation- and rotation-invariant, they were unable to take advantage of spatial information in an image [1]. For the analysis and classification of aerial and satellite images, texture features were commonly used, including Gabor features, co-occurrence matrices and binary patterns [2]. SIFT used the gradient information about important keypoints to describe regions of the image. There are different types of SIFT (sparse SIFT, PCA-SIFT and SURF) which are all highly distinctive and are scale-, illumination- and rotation-invariant [3]. HOG calculates the distribution of the intensities and directions of the gradients of regions in the image, and has had great success at edge detection and identifying shape details in images [4]. Both SIFT and

HOG use features to represent local regions of an image, and therefore reduce their effectiveness by not taking important spatial information into account. To improve these methods, creating global feature representation *bag of visual word* models were introduced [5]. Advances in these models included using pooling techniques such as spatial pyramid matching [6].

Several unsupervised approaches have been explored for remote sensing classification problems. Principal component analysis (PCA) and k-means clustering were successful early methods. More recently, auto-encoders have been used in the area as an unsupervised model, which involve reconstructing an image after forcing it through a bottleneck layer. These unsupervised methods improved on hand-crafted features techniques, but distinct class boundaries were hard to define because the data was unlabeled. For this reason, supervised learning was more attractive, especially for convolutional neural networks (CNNs) from the field of deep learning, which have been responsible for state-of-the-art results in image classification [7,8].

Given the unmatched power of deep learning for image classification, it is natural to investigate the usefulness of CNNs on remote sensing data problems [9,10]. CNN models such as AlexNet [11] and VGG16 [12] have demonstrated their ability to extract relevant informative features that are more discriminative than extracted hand-crafted features. [13,14] used a promising strategy of extracting the CNN activations of the variously scaled local regions, and pooling them together into a bag of local features. Their networks were pre-trained on ImageNet, similar to [15], demonstrating how using a good initialization of parameters can increase network classification accuracy. Other related works show that by avoiding pooling and instead focusing on multi-scale CNN structures, competitive results can be achieved [16].

This type of image classification is very challenging for several reasons. First, although remote-sensing datasets are increasing in size, most are still considered small in deep-learning terms. This means that we often have insufficient training data to obtain high classification accuracy. To have a fair comparison between our method and related work in this area, we set up the experiments in the same manner. This meant that the training dataset had a very limited number of samples, adding to the problem difficulty. To tackle this problem we use *transfer learning*, which uses other data to provide a good initialization point for the network parameters. A second problem is that images from the same class can have very different scales and/or orientation. To address this issue, we apply a standard method from deep-learning image recognition: *data augmentation*. A third problem is that high-resolution satellite images can contain overlapping classes, which can have an inverse effect on classification accuracy [17]. Although the method in [17] has had great success, it is very dependent on how the initial low-level hand-crafted features are extracted, which in turn relies greatly on domain knowledge. Using CNNs to extract the relevant features eliminates the need for domain knowledge and hand-crafted features. A fourth problem is that training a CNN on images can lead to very large networks that are prone to overfitting, and unsuitable for deployment on memory- and energy-limited devices.

A current trend in deep learning is network size reduction, but this often uses computationally expensive supervised learning techniques. We propose a novel unsupervised learning approach, based on k-means clustering, for pruning neural networks of unwanted redundant filters and nodes. We find optimal clusters within the filters/nodes of each layer, and discard those furthest from the center along with all their associated parameters. Our new method, which we call PulseNetOne, combines these techniques with fine-tuning phases to recover from any loss in accuracy. Extensive experiments with various datasets and neural networks were carried out to illustrate the performance and robustness of our proposed pruning technique. We compare it with other state-of-the-art remote-sensing classification algorithms, and experiments show that it significantly outperforms them in classification accuracy and regularization while generating much less complex networks.

The rest of this paper is organized as follows. Section 2 discusses the related work on various methods of remote-sensing image classification. The datasets and CNNs used are described in Section 3,

and we explain our proposed method in Section 4. The results are given in Section 5 as well as their evaluation and discussions. Finally, Section 6 concludes the paper.

2. Related Work

A well-established method that has been very successful for satellite image recognition is the *bag of visual words* (BOVW) approach. This usually involves (i) extracting hand-made features (properties derived from the information present in the image itself), using algorithms like SIFT and HOG; (ii) using a clustering algorithm to group the features, thus creating a BOVW with a defined center; and (3) forming feature representations using histograms, by mapping the learned features onto the nearest cluster center [18–23]. Both the authors of [24,25] have employed this technique to remote-sensing classification. To obtain more meaningful features, spatial pyramids and randomized spatial partitions were used. The authors of [26] used a CNN, originally trained on the ImageNet dataset, with spatial pyramid pooling, and only fine-tuned the fully connected layers of the network. The spatial pyramid was inserted between the convolutional and fully connected layers to automatically learn multi-scale deep features, which are then pooled together into subregions.

To remove the need for domain-knowledge hand-crafted features, unsupervised learning was used to learn basis functions to encode the features. By constructing the features using the training images instead of hand-crafted ones, better discriminative features are learned to represent the data. Some of the more popular unsupervised methods implemented in this research area include k-means, PCA and auto-encoders [17,27,28].

The authors of [29] claimed that an important part of remote-sensing classification was to overcome the problems of within-class diversity and between-class similarity, which are both major challenges. The authors proposed to train a CNN on a new discriminative objective function that imposes a metric learning regularization term on the features. At each training iteration random image samples are used to construct similar and dissimilar pairs, and by applying a constraint between the pairs a hinge loss function is obtained. Unlike our proposed algorithm, which is unsupervised and fast, their method requires several parameters to be selected, which most likely will vary depending on the data, and has a complex training procedure that is relatively inefficient. Their results reinforce the idea that unlike most other approaches that only use the CNN for feature extraction, better performance is achieved by training all the layers within the network.

The authors of [30] designed a type of dual network in which features were extracted from the images based on both the objects and the scene of the image, and fused them together, hence the name FOSNet (fusion of object and scene). The authors trained their network using a novel loss function (scene coherence loss) based on the unique properties of the scene. The authors of [31] fused both local and global features by first partitioning the images into dense regions, which were clustered using k-means. A spatial pyramid matching method was used to connect local features, while the global features were extracted using multi-scale completed local binary patterns which were applied to both gray scale and Gabor filter feature maps. A filter collaborative representation classification approach is used on both sets of features, local and global, and images are classified depending on the minimal approximation residual after fusion. The authors of [32] also used a spatial pyramid idea with the AlexNet CNN as its main structure. The authors find that although AlexNet has shown great success in scene classification and as a feature extractor, because of limited training datasets it is prone to overfitting. By using a technique of side supervision on the last three convolutional layers, and spatial pyramid pooling before the first fully connected layer, the authors claim that this unique AlexNet structure, named AlexNet-SPP-SS, helps to counteract overfitting and improve classification. Our work also shows how both AlexNet and VGG16 are prone to overfitting due to lack of training data, but by pruning redundant filters/nodes we add a strong regularization to the networks, which prevents overfitting and increases model efficiency.

The authors of [33] argue that using a deeper network (GoogLeNet) and extracting features at three stages helps to improve the robustness of the model, allowing low-, mid- and high-level features

to contribute more directly to the classification. Instead of the usual additive approach to pooling features, they show that a product principle works better. Their work includes experiments on Scene15, MIT67 and SUN397 datasets, which we shall also use, and show achieve better accuracy using smaller networks and without pooling. We claim that our approach is also more efficient.

The authors of [16] found that scaling scene images induced bias between training and testing sets, which significantly reduces performance. They proposed scale-specific networks in a multi-scale architecture. They also introduce a novel approach to combine pre-training on both the ImageNet and Places datasets, showing that more accurate classification is achieved. The authors mentioned the idea of redundant features by removing redundancy within the network, making it more efficient and forcing stronger regularization, thus improving generalization. Our proposed method develops this idea.

The authors of [34] combined two pre-trained CNNs in a hybrid collaborative representation method. One side of the hybrid model extracted shared features, while the other extracted class-specific features. They extended and improved their method by using various kernels: linear, polynomial, Hellinger and radial basis function. The authors of [35] also used ImageNet networks pre-trained on VGG16 and Inception-v3 as feature extractors, before introducing a novel 3-layer additional network called CapsNet. The first layer is a convolutional layer that converts the input image into feature maps. The next layer consists of 2 reshape functions along with a squash function that transforms it into a 1-D vector before it enters the final layer, which has a node for each class, and is used for classification. Both these works use pre-trained networks, which our work shows can be significantly reduced in size.

The authors of [36] used VGG16 to extract important features, then used a method based on feature selection and feature fusion to merge relevant features into a final layer for classification. Following the work of the authors of [37] on canonical correlation analysis CCA, [36] improved their work by proposing discriminant correlation analysis, which overcame the limitation of CCA that ignored the relationship between class structures in the data by maximizes the correlation between two feature sets while also maximizing the difference between the classes. The authors of [38] adopted a Semantic Regional Graph model to select discriminant semantic regions in each image. The authors used a graph convolutional network originally proposed by the authors of [39], pre-trained on the COCO-Stuff dataset. This type of classification model showed great promise, especially on the difficult SUN397 data on which it achieved 74% classification accuracy.

The authors of [40] introduced a novel way to tackle limited training data, which causes overfitting in state-of-the-art CNNs such as AlexNet and VGG16. They used transfer learning, but also applied traditional augmentation on the original dataset, and collected images from the internet that were most similar to the desired classes. This greatly increased the number of training examples and helped to prevent overfitting. The authors showed that increasing the number of samples in the training data enables state-of-the-art networks to be trained on small datasets, such as scene-recognition data. Our proposed method can be fine-tuned on the original data alone, with standard augmentation due to the strong regularization our pruning approach imposes on the networks.

The authors of [41] evaluated three common methods of using CNNs for remote-sensing datasets with limited data: training a network from scratch, fine-tuning a pre-trained network, and using a pre-trained network as a feature extractor. The preferred method is training a network from scratch as it tends to generate better features and allows for better control of the network. However, this approach is only feasible when adequate training data is available, otherwise either fine-tuning or feature extraction is more appropriate. The authors found that fine-tuning tends to lead to more accurate classification, especially when combined using a linear support vector machine as the classifier. Although our method uses a pre-trained network, we create more informative features by fine-tuning the full network, while applying our novel pruning approach to create a much smaller and more efficient network that helps to overcome overfitting.

The authors of [42], on which this work is based, proposed an iterative CNN pruning algorithm called *PulseNet* which used an absolute filter/node measurement as the decision metric on which

filters and nodes to prune. Similar to the authors of [42], PulseNetOne pruned all parts of the network but, unlike our proposed method, [42] repeatedly pruned all parts reducing the rate of compression as the network converged to being fully pruned. Also, in this work, we use a more intelligent decision pruning metric based on *k-means*. Both methods extract a smaller, more efficient network, and demonstrate their classification accuracy and computational speed during inference testing.

3. Materials and Methods

3.1. PulseNetOne

Our proposed method PulseNetOne is a CNN pruning approach that takes a pre-trained network, and fine-tunes it with the dataset under analysis. PulseNet was the name of the original work, where pruning was performed iteratively, whereas this approach uses a single iteration to prune the network; hence the name PulseNetOne. When the validation loss has converged on the dataset, PulseNetOne prunes each layer in the network using a k-means based method. While other works in this area use the L1-norm or similar metrics to prune parts of the network, we show that by using a more intelligent way of determining which filters/nodes are redundant, we achieve an extremely efficient CNN that can be used for real-time inference.

In each convolutional layer, w, x, y, z represents the tunable parameters, where w and x are the filter's width and height respectively, y the inward number of filters and z the outward number of filters. In the fully-connected layer the inward and outward nodes are represented by m and n respectively. If the layer being pruned is the first convolutional layer then the inward pruning index list is the list of channels in the image (for an RGB image this would be $[0, 1, 2]$). For all other layers the inward pruning index list is the outward pruning index list of the previous layer. The outward pruning indices are found by running a k-means algorithm with all possible k values between 1 and the number of filters/nodes in the layer. The sum of squares (the Euclidean distance between a filter/node and the cluster center) within each cluster is calculated and the algorithm continues to run until it either; finds the minimum distortion or reaches the maximum number of iteration (300). The distortion is the Euclidean distance between the a filter/node and the center of the cluster it lays in. The k-means algorithm, reinitializes 5 times, in an attempt to reduce the possibility of the algorithm only finding a local minimum. The distortion and its corresponding k value are recorded. Algorithm 1 determines the optimal number of filters/nodes to retain. It is based on an automated way of finding the elbow-point on the graph of all possible k values. It finds where an albow point occurs, and then calculates its strength. After it has looked at all k values, it then returns the optimal k based on the one with the maximum value. The same process is followed for the fully-connected layers, with the first fully connected layer's inward nodes being the last convolutional output filters.

Algorithm 2 loops through the layers of the network, pruning each layer in an unsupervised manner, to extract a smaller and more efficient network. The relevant filters/nodes extracted by PulseNetOne are re-used to create a smaller network, and the network is fine-tuned on the dataset being analysed to restore lost accuracy. We experimented with using the centroid of each cluster as the filter/node to keep, but failed to regain the accuracy lost in the pruning. Resetting the parameters associated with the center filter/node (batch-normalization γ and β), along with its bias, to their initial states did not achieve the desired results. Nor did inserting their values within the filter/node vector to be calculated using k-means. So instead of using the k-means centroids we use the filter/node that is closest to each centroid (a fast approximation to the medoid). This allows PulseNetOne to re-use a filter/node with its biases and other associated parameters, which are already trained on the dataset being analysed.

3.2. Benchmark Remote-Sensing Image-Scene Datasets

We evaluated our proposed PulseNetOne method on 4 benchmark remote-sensing datasets and 2 scene-recognition datasets (MIT 67 and SUN397), seen in Table 1. These range in difficulty in terms of

the scale of scene classes, images per class, and diversity within the data. This means that on some datasets like UC Merced [43] 97%–99% classification accuracy is easily attainable using CNNs and could be considered the *MNIST* dataset of remote-sensing research (an early dataset now considered trivial). Therefore, although we show our proposed method on these datasets, it is mainly to compare to other work. To demonstrate the potential of PulseNetOne we also use harder benchmark datasets such Remote-Sensing Image-Scene Classification (RESISC) [44] and SUN397 [45]. The datasets are:

Table 1. Comparison of the six remote sensing datasets used in this work.

Dataset	# of Classes	Ĩmages per Class	Total Images	Train/Test Split	Image Sizes	Year
Scene-15 [24]	15	210-410	4485	100 imgs/rest	Multi Scales	2006
MIT-67 [46]	67	101-734	15620	80 imgs/rest	Multi Scales	2009
UC Merced [25]	21	100	2100	80%/20%	256 × 256	2010
SUN397 [45]	397	100-2361	10000	40 imgs/rest	Multi Scales	2010
NWPU-RESISC45 [44]	45	700	31500	20%/80%	256 × 256	2016
AID [47]	30	220-420	10000	20%/80%	600 × 600	2016

- The Aerial Image Dataset (AID) [47] contains 10,000 images with 30 classes including commercial, forest and church. The image resolution was 600 × 600 pixels but they were resized to 256 × 256 to compare with other works. The dataset was split with 50% randomly selected for the training set and 50% for the test set.
- The MIT 67 dataset [46] has 67 classes representing types of indoor scene. It has 15,620 RGB images which were resized to 224 × 224 following the literature. The dataset was randomly divided into 80 samples for training and the remainder used for testing. To determine when the networks have converged, a validation dataset was needed. This was created using the training data with 60 images kept in the training set and 20 used for a validation set. Because of the large number of targets and some similar class types, this is one of the more difficult remote sensing classification tasks.
- The NWPU-RESISC45 dataset [29] consists of 31,500 remote sensing images with 45 categories. The images are of size 256 × 256 and there are 700 per class. As suggested in the literature, 20% of the samples within each class are used as the training dataset and the remaining 80% as the test set.
- A small but popular dataset is Scene15 [24], made up of natural and indoor categories with each class having between 210 and 410 examples. To compare with results in the literature, 100 random samples from each class are used for training and the rest for testing. The training dataset is split further into 80 images per class for training and 20 for validation. Although this is the oldest dataset used in this work, it still is a good test for the robustness of our method, and has a great deal of related work for comparison.
- The SUN397 dataset [45] is a much larger dataset made up of 108,754 images with 397 classes, ranging from 100 to 2361 examples per class, and including natural, indoor and fabricated images. Only 40 samples are used for training of each class, 10 images per class for the validation set and the rest for the test set. For this work, and as reported by others, this was the most challenging remote-sensing dataset for two main reasons: the large number of classes and the small training set.
- The UC Merced Land-Use dataset (UCM) [25] has 2100 scene images taken from above, separated into 21 classes with 100 examples of 256 × 256 pixels in each class. The dataset was constructed from aerial orthoimagery in various US regions including Dallas, Tampa and Las Vegas, with categories including runway, golf course and different degrees of residential density.

Due to the limited number of training samples, to compute the final model accuracy the validation set is merged with the training set to recreate the original training data, and the model is further trained for a few epochs [33]. The experiments are validated using 5-fold cross validation, randomly selecting the training and testing data splits at each fold. All images were resized to 256 × 256 where necessary.

Algorithm 1 Get optimal k clusters

Let $WCSS \rightarrow$ sum of squares within clusters
given input M
for k in $1 \rightarrow \text{len}(M)$
    randomly choose k vectors V as the initial centers of k clusters C
    Repeat
        (re)assign each V to C to which it is most similar, based on mean value of V within C
        update C means
    Until *no change*
    store sum of squares within clusters WCSS
calculate $D1 = WCSS_{i+1} - WCSS_i, \quad \forall \quad i \in 1 \rightarrow \text{len}(WCSS) - 1$
calculate $D2 = D1_{i+1} - D1_i, \quad \forall \quad i \in 1 \rightarrow \text{len}(D1) - 1$
calculate $\sigma = D2 - D1$
if $\sigma > 0$
    store in cluster index $indx$
find $C = \max(indx)$
for center in C
    find v = vector closest to center
return indices of v

Algorithm 2 PulseNetOne K-means Algorithm

initialize $p_{in} = [0,1,2]$
given layers L
$\forall \quad l \in L$
    if l = convolution layer
        given filter F represented by w, x, y, z
        subset $(w_{(:)}, x_{(:)}, y_{(p_{in})}, z_{(:)})$
        reshape filter $w, x, y, z \rightarrow z, w * x * y$
        p_{out} =Get optimal k clusters (Algorithm 1)
        subset $(z_{(p_{out})}, w_{(:)} * x_{(:)} * y_{(:)})$
    if l = fully-connected layer
        given node N represented by m, n
        subset $(m_{(p_{in})}, n_{(:)})$
        reshape node $m, n \rightarrow n, m$
        p_{out} =Get optimal k clusters (Algorithm 1)
        subset $(n_{(p_{in})}, m_{(:)})$
    $p_{in} \leftarrow p_{out}$

3.3. Convolutional Neural Networks

To evaluate the effectiveness of our proposed method, we run experiments using two state-of-the-art image recognition CNNs (AlexNet and VGG16) in various states, on all six of the image datasets. The experiments are: train from scratch, apply transfer learning with models trained on Imagenet, fine-tune pre-trained Imagenet models, and finally prune the fine-tuned models, which are then further fine-tuned to regain accuracy.

For transfer learning with models on Imagenet, the fully connected layers are removed and new ones added. The convolutional layers are frozen and only the fully connected layers are trained using the Adam optimizer with learning rate 10×10^{-5}. The fine-tuned pre-trained Imagenet models are the same as the transferred versions, except that all layers of the network are fine-tuned on the desired

dataset. Finally, the pruned model is extracted from the fine-tuned model. Our proposed PulseNetOne method removes redundant filters/nodes from the fine-tuned model, leaving a pruned model.

The first network used was the AlexNet network, proposed by the authors of [11], which was a CNN based on the LeNet network of 1998. It was deeper and wider than LeNet, consisting of 5 convolutional layers followed by 2 dense fully connected layers. They were one of the first to take advantage of graphics processing units (GPUs): in fact their network was trained using two GPUs in parallel. The AlexNet version used in this work linked the separate pairs of convolutional blocks into one larger block. As well as establishing CNNs in the area of image recognition, [11] introduced ReLU activation function to CNNs, and dropout as a method for avoiding overfitting. In this work we take advantage of both.

The other network used was the VGG16 network, introduced by the authors of [12], which had 16 layers ranging in number of filters from 64 to 512, with 2 fully connected dense layers of 4096 nodes. Unlike AlexNet they used 3×3 filters in all convolutional layers, with 5 layers of 2×2 maxpooling. Their main contribution was to demonstrate the importance of network depth for classification performance. One downside to this improvement was that it was more expensive than previous networks such as AlexNet, in terms of both memory and speed.

3.4. Experimental Design

The proposed method was implemented in Python using the TensorFlow deep-learning framework. The training and pruning phases were performed on a RTX2080 Ti NVIDIA graphics processing unit (GPU), while the inference stage was evaluated on both a RTX2080 GPU and an INTEL I7-8700K CPU with 32GB RAM.

The six remote-sensing datasets had variously sized images which, for comparison with related work, were resized to 256×256 pixels. We used standard data augmentation including random horizontal flipping, random adjustment of brightness and random increase/decrease of the image contrast. We also performed random cropping of the training dataset down to 224×224, and the test dataset was centrally cropped to 224×224. All images were then standardized by per-color mean and their standard deviation. The optimizer used for training the network was Stochastic gradient descent (SGD), in conjunction with a stepwise decay learning rate schedule. The initial learning rate for both training and fine-tuning was 0.1, reduced by a factor of 10 when no decrease in loss on the validation set is detected for 20 epochs. The minimum learning rate used was 0.0001, and once there has been no improvement in the loss at that rate for 20 epochs the network is considered to have converged. A training mini-batch of 32 was used, while during the inference stage single test samples were used as the inputs to replicate a real-world scenario.

4. Results

PulseNetOne was applied to the AID dataset and its performance compared with other state-of-the-art results. Table 2 shows that training the networks on the target dataset from scratch yielded poor results: 61.24% and 56.67% on AlexNet and VGG16 respectively. The accuracy of both networks improved greatly when using transfer learning and fine-tuning the transferred model. For transfer learning the weights and parameters of networks trained on ImageNet, on both AlexNet and VGG16, were re-initialized and used as a starting point for the weights of our networks. The fully-connected layers of the pre-trained networks were removed and replaced by fully-connected layers of the same size initialized using a *He normal* weight distribution, with the number of final layer outputs being the number of classes in the dataset.

PulseNetOne takes the fine-tuned network and, as described in Section 3.1, prunes the original network to a much smaller version, which not only reduces the storage size and number of floating point operations per second (FLOPs), but also improves classification accuracy. The accuracy of AlexNet improves by nearly 5.5% and VGG16 by 3.5%. The confusion matrices of both networks (available on request) show that very few errors were made, and as they were quite randomly

distributed they might simply be a poor representation of the class within wrongly-classified images. The precision, recall and F1 scores of both networks are in their descriptions for further comparison.

Figure 1 shows the number of filters pruned in each layer of the networks. As expected, the layers pruned most are the fully-connected layers, as it is well documented that these are over-parameterized. It is interesting to see that the first and last few convolutional layers are pruned more than the intermediate layers. A reason for this is that the first few layers are edge detectors and filters based on colour and shapes which can contain a lot of duplicate or similar filters, while the last convolutional layers are more class-related and, because the networks were pre-trained on the 1000-class ImageNet dataset, these layers contained many redundant filters.

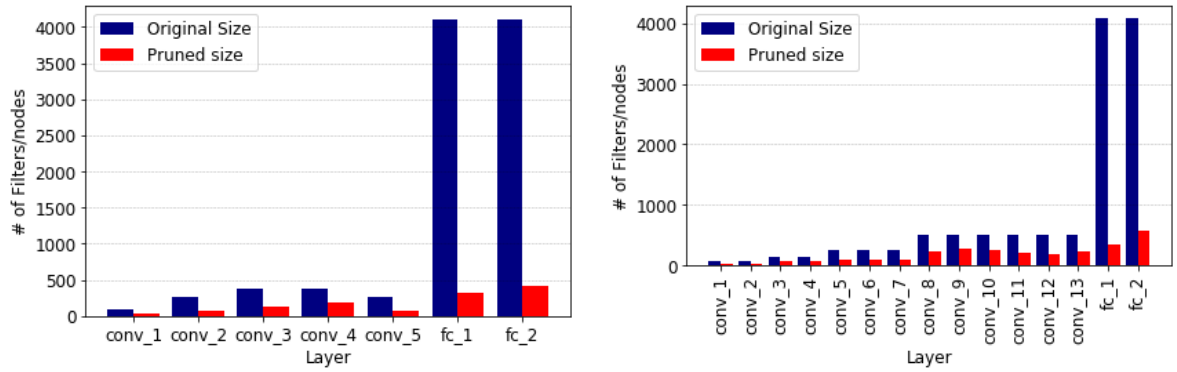

Figure 1. Comparison of layers between original and pruned version of both AlexNet and VGG16 on the AID dataset.

Table 2. Overall accuracies and standard deviations (%) of different types of CNNs methods along with PulseNetOne on the AID dataset. The entries in bold show the method with the best classification for the network, while the percentage of the original network is highlighted in red.

Method	Network Structure	# Parameters	# FLOPs	Accuracy
AlexNet				
Scratch	96 – 256 – 384 – 384 – 256 – 4096 – 4096	58404254 (100%)	116789320 (100%)	61.24 ± 1.53
Transferred	-	-	-	91.17 ± 0.10
Fine tuned	-	-	-	93.51 ± 0.12
PulseNetOne	36 – 66 – 135 – 180 – 79 – 317 – 409	1544061 (2.64%)	3085626 (2.64%)	**98.91 ± 0.07**
VGG16				
Scratch	64 – 64 – 128 – 128 – 256 – 256 – 256 – 512 –512 – 512 – 512 – 512 – 4096 – 4096	134383454 (100%)	268742032 (100%)	56.67 ± 0.31
Transferred	-	-	-	93.64 ± 0.26
Fine tuned	-	-	-	96.11 ± 0.19
PulseNetOne	23 – 28 – 60 – 59 – 90 – 95 – 103 – 228 –267 – 243 – 203 – 174 – 230 – 354 – 570	6942627 (5.17%)	13879756 (5.16%)	**99.77 ± 0.06**

The barcharts in Figure 2 clearly show that, although in most cases the theoretical improvements are not reached (except for the GPU timings for both networks, and the GPU energy for AlexNet), the results come close in most cases. Comparing our work with state-of-the-art results in Table 3, it can be seen that our approach using VGG16 achieves the best classification accuracy with 99.77%, and our AlexNet version ranks in second place with 98.91%, which outperforms both Discriminative CNNs VGG16 [29] and GCFs+LOFs [48] by nearly 3%.

Next the dataset MIT67 is analysed, on which (similarly to the AID dataset) CNNs achieve poor classification accuracy when trained from scratch: 32.31% accuracy on AlexNet and 26.57% on VGG16. The explanation is believed to be the lack of training samples: a deep learning network has millions of parameters to tune and is therefore quite data-hungry. Table 4 shows that simply transferring learning was not as effective as on the previous dataset, reaching a maximum accuracy of nearly 70%, but after fine-tuning on the targeted dataset, it reached a more reasonable 92.74%. PulseNetOne

reduces AlexNet down to 2.28% and VGG16 down to 8.26% of their original sizes, and improves their performances to 95.83% and 96.68% respectively.

Table 3. A comparison between state-of-the-art and PulseNetOne results on the AID data set. The entries in bold show the method with the best classification for the network.

Method	Year	Accuracy
CaffeNet [47]	2017	89.53
VGG-VD-16 [47]	2017	89.64
Fusion by addition [36]	2017	91.87
Discriminative CNNs AlexNet [29]	2018	94.47
Two-Stream Fusion [49]	2018	94.58
VGG16-CapsNet [35]	2019	94.74
Discriminative CNNs GoogLeNet [29]	2018	96.22
Inception-v3-CapsNet [35]	2019	96.32
GCFs+LOFs [48]	2018	96.85
Discriminative CNNs VGG16 [29]	2018	96.89
AlexNet-PulseNetOne	2020	**98.91**
VGG16-PulseNetOne	2020	**99.77**

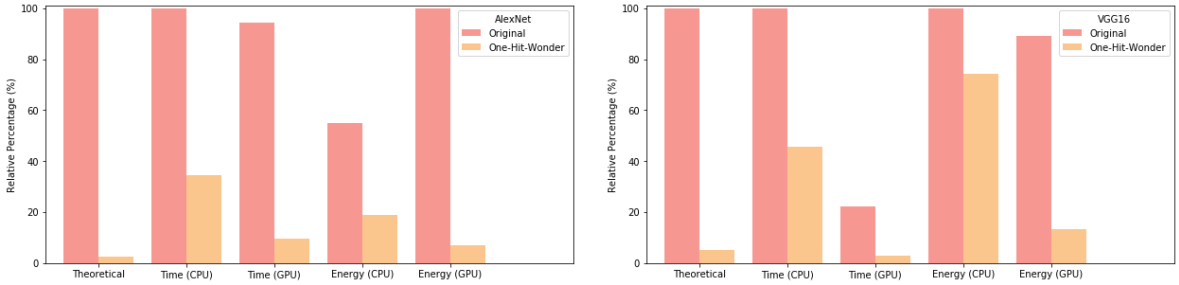

Figure 2. Barcharts for AlexNet and VGG16, illustrating the difference between the theoretical, CPU and GPU efficiency with respect to their computational speed and energy consumed on the AID dataset.

Table 4. Overall accuracies and standard deviations (%) of different CNNs methods along with PulseNetOne on the MIT67 dataset. The entries in bold show the method with the best classification for the network, while the percentage of the original network is highlighted in red.

Method	Network Structure	# Parameters	# FLOPs	Accuracy
AlexNet				
scratch	96 − 256 − 384 − 384 − 256 − 4096 − 4096	58555843 (100%)	117092424 (100%)	32.31 ± 2.36
Transferred	-	-	-	63.45 ± 0.61
Fine-tuned	-	-	-	91.84 ± 1.28
PulseNetOne	35 − 118 − 132 − 133 − 67 − 317 − 204	1338355 (2.29%)	2674572 (2.28%)	**95.83 ± 0.11**
VGG16				
scratch	64 − 64 − 128 − 128 − 256 − 256 − 256 − 512− 512 − 512 − 512 − 512 − 512 − 4096 − 4096	134535043 (100%)	269045136 (100%)	26.57 ± 1.38
Transferred	-	-	-	69.92 ± 2.08
Fine-tuned	-	-	-	92.74 ± 1.31
PulseNetOne	28 − 15 − 32 − 36 − 104 − 100 − 91 − 143− 154 − 163 − 173 − 144 − 220 − 826 − 752	11111636 (8.26%)	22217192 (8.26%)	**96.68 ± 0.62**

The confusion matrices for both networks (available on request) show that most mistakes were made when distinguishing between the bathroom and bedroom, and the grocery store and toy store classes. The precision, recall and F1 scores of both networks are in their descriptions for further transparency, and can be seen to be between 95.89% and 96.92%. Details of how PulseNetOne pruned the layers of the networks are shown in Figure 3, and agree with the analysis of the AID dataset. However, the VGG16 network retained more nodes in the fully-connected layers, which could be caused by the MIT67 dataset having more than twice the number of target classes.

Table 5 compares PulseNetOne to the related work in this area, and it can be seen that both networks pruned by PulseNetOne outperform the state-of-the-art by over 6%. FOSNet CCG [30] and SOSF+CFA+GAF [50] were the current best published results on the MIT67 dataset, achieving 90.37% and 89.51% respectively, but were significantly beaten by PulseNetOne. Figure 4 shows that AlexNet almost achieved its theoretical performance on all experiments except for CPU inference timing, while the pruned network was approximately 3× faster than the original network. VGG16 results were more mixed, with the CPU timing beating the theoretical result, but the CPU energy usage being quite high, though still slightly less than the original network structure.

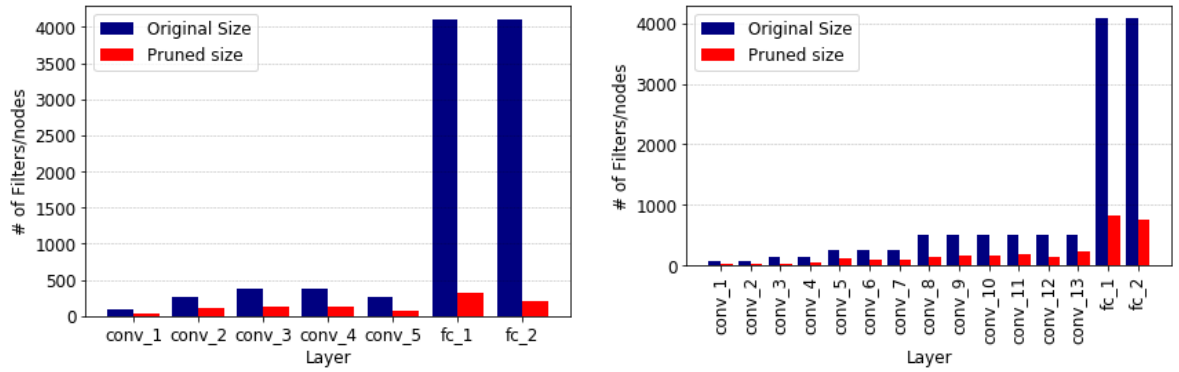

Figure 3. Comparison of layers between original and pruned version of both AlexNet and VGG16 on the MIT67 dataset.

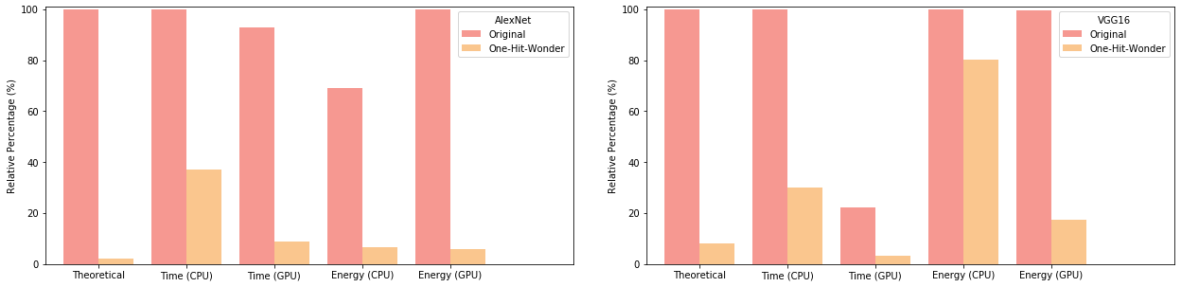

Figure 4. Barcharts for AlexNet and VGG16, illustrating the difference between the theoretical, CPU and GPU efficiency with respect to their computational speed and energy consumed on the MIT67 dataset.

The NWPU-RESISC45 dataset accuracy was only able to score 27.11% on AlexNet and 17.87% on VGG16 when trained from scratch. Table 6 shows that transfer learning boosted their performances up to approximately 79% accuracy, while fine-tuning increased both to approximately 84%. PulseNetOne was able to increase their classification accuracy by over 10%, with AlexNet scoring 94.65% and VGG16 94.86%. This was the result of network pruning reducing overfitting: AlexNet was reduced by 67× and VGG16 by 33×.

The confusion matrices for both networks (available on request) show that both networks found it hard to distinguish between the freeway and railway, medium and dense residential, and meadow and forest classes. Other publications have commented on these classes being difficult to separate also. Again, the precision, recall and F1 scores of both networks are given in their descriptions, and can be seen to be between 94.65% and 94.89%. The way in which PulseNetOne pruned the layers of the networks, shown in Figure 5, is more similar to that of the AID dataset than the MIT67 dataset, possibly because of its 45 classes.

Figure 6 shows that once again AlexNet achieved close to its theoretical performance on all experiments except for the CPU inference timing. The VGG16 results were not quite as impressive, with the GPU timing beating the theoretical result, while the CPU energy usage was close to that of the

original network. It can be seen from Figure 6 that in the other experiments the pruned network is much more efficient than the original.

Table 5. A comparison between state-of-the-art and PulseNetOne results on the MIT67 data set. The entries in bold show the method with the best classification for the network.

Method	Year	Accuracy
Otc and HOG [51]	2014	47.33
AlexNet fine-tuned on Imagenet [52]	2014	56.79
AlexNet fine-tuned on Place205 [52]	2014	68.24
Hybrid-CNN [52]	2014	70.80
GoogLeNet fine-tuned on Imagenet [53]	2014	72.31
DSFL CNN [14]	2017	76.23
GoogLeNet fine-tuned on Place205 [53]	2014	77.54
DSP [54]	2014	78.28
InterActive [55]	2016	78.65
G-MS2F (ADD) [33]	2017	79.18
G-MS2F (Prod) [33]	2017	79.63
SDO [29]	2018	86.76
MP [56]	2017	86.90
Sparse Representation [57]	2017	87.22
MFAFVNet+Places [58]	2017	87.97
SRG [38]	2019	88.13
SOSF+CFA+GAF [50]	2018	89.51
FOSNet CCM-CCG [30]	2019	90.30
FOSNet CCG [30]	2019	90.37
AlexNet-PulseNetOne	2020	**95.83**
VGG16-PulseNetOne	2020	**96.68**

Table 6. Overall accuracies and standard deviations (%) of different CNNs methods along with PulseNetOne on the NWPU-RESISC45 dataset. The entries in bold show the method with the best classification for the network, while the percentage of the original network is highlighted in red.

Method	Network Structure	# Parameters	# FLOPs	Accuracy
AlexNet				
scratch	96 – 256 – 384 – 384 – 256 – 4096 – 4096	58465709 (100%)	116912200 (100%)	27.11 ± 2.63
Transferred	-	-	-	79.86 ± 0.18
Fine-tuned	-	-	-	83.29 ± 0.25
PulseNetOne	30 – 76 – 115 – 62 – 58 – 257 – 231	850336 (1.45%)	1698932 (1.45%)	**94.65 ± 0.95**
VGG16				
scratch	64 – 64 – 128 – 128 – 256 – 256 – 256 – 512– 512 – 512 – 512 – 512 – 512 – 4096 – 4096	134444909 (100%)	268864912 (100%)	17.87 ± 3.47
Transferred	-	-	-	78.85 ± 0.16
Fine-tuned	-	-	-	84.36 ± 0.22
PulseNetOne	21 – 20 – 53 – 45 – 88 – 109 – 74 – 230– 165 – 194 – 200 – 199 – 137 – 300 – 268	4074009 (3.03%)	8143738 (3.03%)	**94.86 ± 1.03**

PulseNetOne again beats the current state-of-the-art on the NWPU-RESISC45 dataset, as seen in Table 7, by just over 2%. Inception-v3-CapsNet [35] and Triple networks [59] achieve 92.60% and 92.33% respectively, which is quite close to our results, but it should be noted that PulseNetOne creates an extremely efficient version of the networks, while both the related works approaches use complex networks that increase computational expense.

The accuracy achieved on Scene15 dataset, when trained from scratch, was quite reasonable when compared to the previous datasets, with AlexNet scoring 77.91% and VGG16 scoring 79.69%. Transfer learning increased classification accuracy by 10–12%, while fine-tuning further increased it by 2–4% as seen in Table 8. PulseNetOne was able to increase the classification accuracy of AlexNet to 97.96%

and VGG16 to 97.48%. AlexNet was reduced to 1.3% and VGG16 to 3.04% of their original sizes. The confusion matrices of both networks (available on request) show no particular pattern in their errors, with both networks making random misclassifications. The precision, recall and F1 scores of both networks are given in their descriptions, and can be seen to be between 97.46% and 98%.

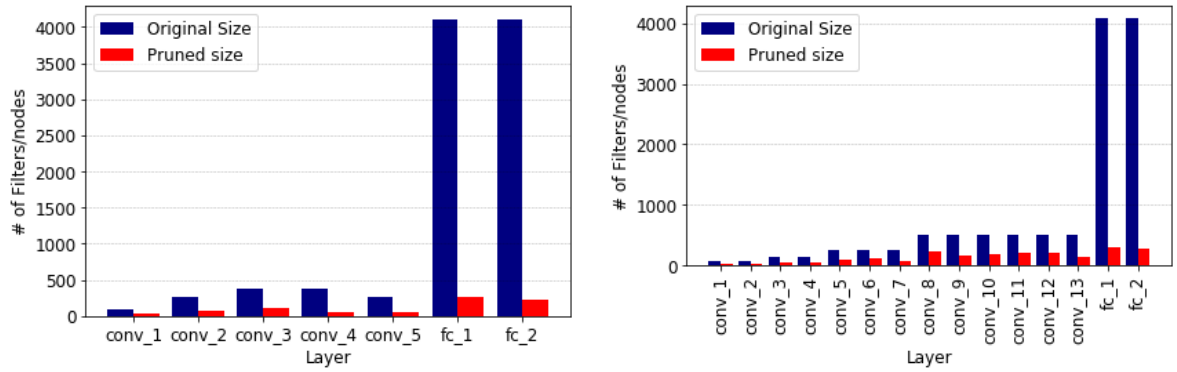

Figure 5. Comparison of layers between original and pruned version of both AlexNet and VGG16 on the NWPU-RESISC45 dataset.

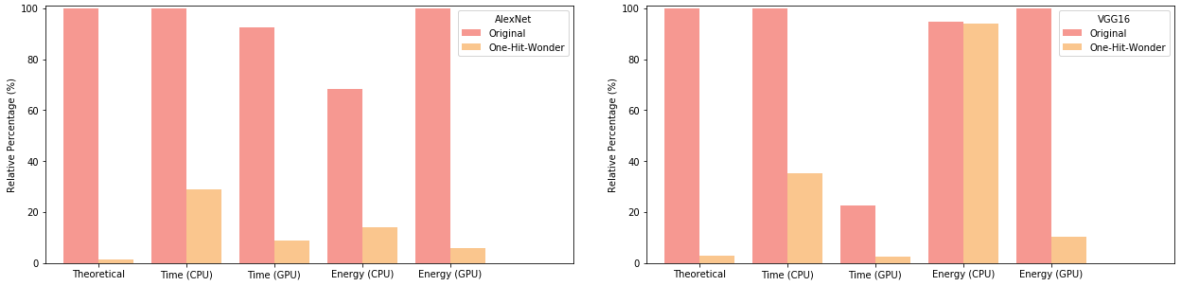

Figure 6. Barcharts for AlexNet and VGG16, illustrating the difference between the theoretical, CPU and GPU efficiency with respect to their computational speed and energy consumed on the NWPU-RESISC45 dataset.

Table 7. A comparison between state-of-the-art and PulseNetOne results on the NWPU-RESISC45 dataset. The entries in bold show the method with the best classification for the network.

Method	Year	Accuracy
Two-Stream Fusion [49]	2018	83.16
Fine-tuned CNNs AlexNet [44]	2017	85.16
Fine-tuned CNNs GoogLeNet [44]	2017	86.02
Discrimative CNNs AlexNet [29]	2018	87.24
VGG16-CapsNet [35]	2019	89.18
Fine-tuned CNNs VGG16 [44]	2017	90.36
Discrimative CNNs GoogLeNet [29]	2018	90.49
Discrimative CNNs VGG16 [29]	2018	91.89
Triple networks [59]	2017	92.33
Inception-v3-CapsNet [35]	2019	92.60
AlexNet-PulseNetOne	2020	**94.65**
VGG16-PulseNetOne	2020	**94.86**

The layers of both networks, as shown in Figure 7, are pruned in the same pattern as with the other datasets, fully-connected layers being heavily pruned, along with the beginning and ending of the convolutional layers, while the intermediate convolutional layers are less pruned.

Figure 8 shows that on AlexNet the pruned network performs better on the GPU, but for real-world situations where a CPU would be more commonly used for analysis, the pruned network easily outperforms the original structure in all cases. On VGG16 the CPU energy usage are

unimpressive, but are almost twice as energy efficient as the original network. Following the same analysis as for the previous datasets, we compare PulseNetOne to the current state-of-the-art on the Scene15 dataset, shown in Table 9. The Dual CNN [16] was state-of-the-art since 2016, with a classification accuracy of 95.18%, the next in line being G-MS2F [33] with 92.90%. PulseNetOne achieves almost a 3% greater accuracy, with AlexNet beating VGG16 on this dataset with 97.97%. Again, PulseNetOne's closest competition uses 2 deep learning CNNs which is much less efficient.

Table 8. Overall accuracies and standard deviations (%) of different CNNs methods along with the proposed PulseNetOne on the Scene15 dataset. The entries in bold show the method with the best classification for the network, while the percentage of the original network is highlighted in red.

Method	Network Structure	# Parameters	# FLOPs	Accuracy
AlexNet				
scratch	96 – 256 – 384 – 384 – 256 – 4096 – 4096	58342799 (100%)	116666440 (100%)	77.91 ± 0.85
Transferred	-	-	-	87.52 ± 0.38
Fine-tuned	-	-	-	91.68 ± 0.72
PulseNetOne	36 – 90 – 168 – 53 – 32 – 301 – 274	759853 (1.30%)	1517776 (1.30%)	**97.96 ± 0.62**
VGG16				
scratch	64 – 64 – 128 – 128 – 256 – 256 – 256 – 512– 512 – 512 – 512 – 512 – 512 – 4096 – 4096	134321999 (100%)	268619152 (100%)	79.69 ± 1.27
Transferred	-	-	-	89.75 ± 1.01
Fine-tuned	-	-	-	92.84 ± 0.48
PulseNetOne	10 – 22 – 36 – 36 – 65 – 68 – 76 – 109– 168 – 198 – 167 – 167 – 169 – 305 – 257	4079549 (3.04%)	8155378 (3.04%)	**97.48 ± 0.81**

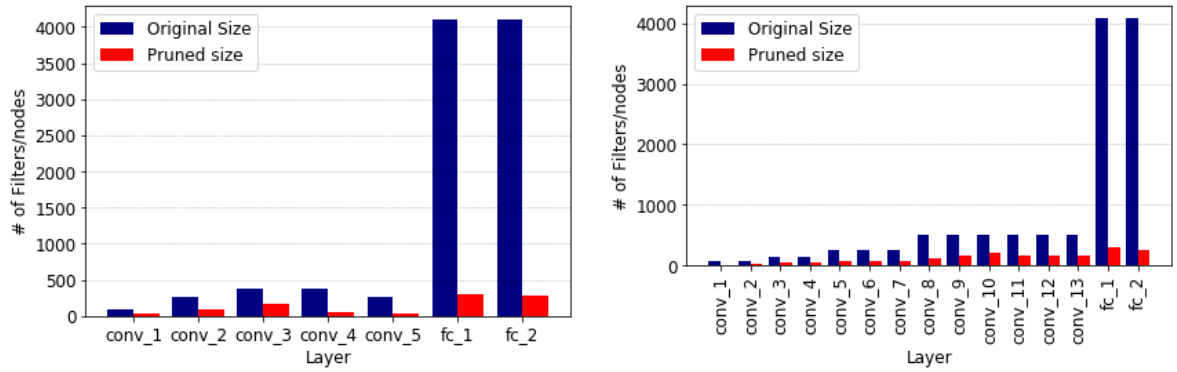

Figure 7. Comparison of layers between original and pruned version of both AlexNet and VGG16 on the Scene15 dataset.

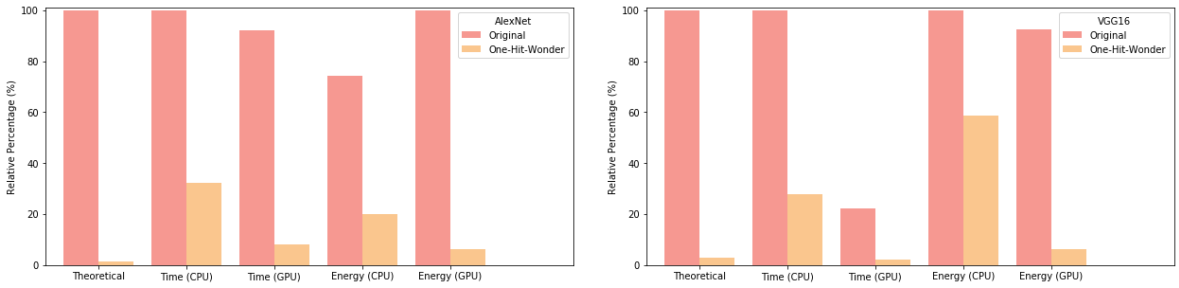

Figure 8. Barcharts for AlexNet and VGG16, illustrating the difference between the theoretical, CPU and GPU efficiency with respect to their computational speed and energy consumed on the Scene15 dataset.

The SUN397 dataset was the most difficult dataset to train from scratch, possible because of its large number of classes and its limited amount of training data: AlexNet reached 21.24% and VGG16 15.41% accuracy. Transfer learning with the ImageNet dataset helped increased AlexNet's accuracy to

42.91% and VGG16 to 41.51%. Fine-tuning all the layers for either network had less effect than with the other datasets, only improving AlexNet to 49.89% and VGG16 to 50.29%. However, PulseNetOne was able to pass 80% classification accuracy on both networks, as seen in Table 10: AlexNet reached 82.11% accuracy and VGG16 84.32%.

Table 9. A comparison between state-of-the-art and PulseNetOne results on the SCENE15 data set. The entries in bold show the method with the best classification for the network.

Method	Year	Accuracy (%)
AlexNet fine-tuned on Imagenet [52]	2014	84.23
Otc and HOG [51]	2014	84.37
LGF [31]	2016	85.80
GoogLeNet fine-tuned on Imagenet [53]	2014	91.12
AlexNet fine-tuned on Place205 [52]	2014	90.19
Hybrid-CNN [52]	2014	91.59
DSP [54]	2014	92.16
GoogLeNet fine-tuned on Place205 [53]	2014	92.16
G-MS2F (ADD) [33]	2017	92.70
DSFL CNN [14]	2017	92.81
G-MS2F (Prod) [33]	2017	92.90
Dual CNN [16]	2016	95.18
VGG16-PulseNetOne	2020	**97.48**
AlexNet-PulseNetOne	2020	**97.96**

Table 10. Overall accuracies and standard deviations (%) of different CNNs methods along with PulseNetOne on the SUN397 dataset. The entries in bold show the method with the best classification for the network, while the percentage of the original network is highlighted in red.

Method	Network Structure	# Parameters	# FLOPs	Accuracy
AlexNet				
scratch	96 – 256 – 384 – 384 – 256 – 4096 – 4096	59907853 (100%)	119795784 (100%)	21.24 ± 1.98
Transferred	-	-	-	42.91 ± 1.71
Fine-tuned	-	-	-	49.89 ± 1.18
PulseNetOne	35 – 103 – 176 – 160 – 41 – 242 – 264	1105768 (1.85%)	2208708 (1.84%)	**82.11 ± 2.48**
VGG16				
scratch	64 – 64 – 128 – 128 – 256 – 256 – 256 – 512– 512 – 512 – 512 – 512 – 512 – 4096 – 4096	135887053 (100%)	271748496 (100%)	15.41 ± 1.75
Transferred	-	-	-	41.51 ± 3.84
Fine-tuned	-	-	-	50.29 ± 2.89
PulseNetOne	16 – 27 – 35 – 38 – 68 – 69 – 72 – 118– 160 – 164 – 161 – 10 – 48 – 236 – 208	1562544 (1.15%)	3121450 (1.15%)	**84.32 ± 2.94**

Confusion matrices for both networks (available on request) showed no real surprises in either of them, similar classes being mis-classified. The PulseNetOne-pruned AlexNet model had a precision score of 85.60%, recall score 82.45% and F1-score 83.24%, and the VGG166 version had precision score 86.80%, recall score 84.29% and a F1-score 84.93%. The layers of both networks, as seen in Figure 9, are similar to previous results, but show that AlexNet's last convolution layer was heavily pruned: slightly surprising given the dataset's large number (397) of classes. A possible explanation is that the classes in the ImageNet dataset, on which the model was initially trained, were quite different to the classes on the target dataset SUN397. The PulseNetOne-pruned VGG16 also had a convolutional layer that was heavily pruned (the 12*th* layer or second-to-last layer), which reaffirms our hypothesis.

Figure 10 shows that the PulseNetOne version of both networks use considerably less energy and are in all cases noticeably faster at inference time. Table 11 shows previous state-of-the-art results, and our method advances the best by approximately 5%. SOSF+CFA+GAF [50] achieves the closest to our results, but whereas we use an input image of size 256 × 256, their method uses an input size of

608 × 608 which is signifcant larger and therefore more computationally expensive. FOSNet [30] is once again close to the state-of-the-art, with 77.28%.

The final dataset is UC Merced, which (as mentioned earlier) is relatively easy to classify accurately. VGG16 and AlexNet achieve 75.01% and 83.25% respectively, while transfer learning applied only to the new fully-connected layers resulted in accuracies of approximately 95%. Further fine-tuning increased the accuracies of both networks to just over 98%, and the current state-of-the-art uses fine-tuning with a Support Vector Machine as the classifier. PulseNetOne added 1.5% accuracy to both networks, and though this is small it makes our proposed method state-of-the-art. Table 12 shows that VGG16 achieved classification accuracy 99.69%, and AlexNet 99.82%. We attribute this improvement to the high degree of pruning reducing overfitting: the AlexNet parameters and FLOPS were reduced by 30×, and those of VGG16 by 71.5×.

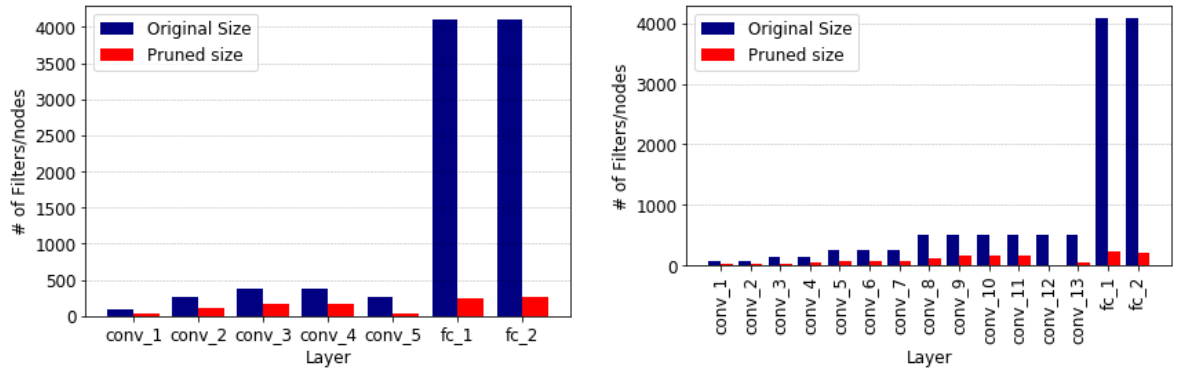

Figure 9. Comparison of layers between original and pruned version of both AlexNet and VGG16 on the SUN397 dataset.

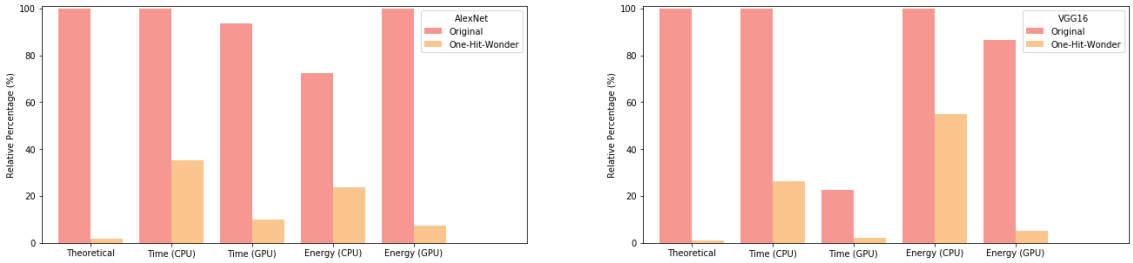

Figure 10. Barcharts for AlexNet and VGG16, illustrating the difference between the theoretical, CPU and GPU efficiency with respect to their computational speed and energy consumed on the SUN397 dataset.

The confusion matrices of both networks (available on request) show that between both networks there was only 3 misclassifications. The precision, recall and F1 scores of both networks are in their descriptions, and can be seen to be 99.36–99.70%. As expected, the GPU performance was close to theoretical, while CPU times were significantly better on the pruned networks compared to the original networks, as shown in Table 12.

Figure 11 shows that the second-last layer in VGG16 is again the most pruned, as in the SUN397 dataset, with the other layers being pruned in the same pattern as the other datasets. Figure 12 shows that the GPU was more energy efficient and also had a faster inference time, but the PulseNetOne versions of the networks were both faster and consumed less energy on the CPU than the original networks. PulseNetOne slightly outperforms the current state-of-the-art on the UC Merced dataset by almost 0.5%, as shown in Table 13. Inception v3 CapsNet [35] had, once again, one of the best accuracies with 99.05%, narrowly beaten by Fine-tuned GoogLeNet with SVM [41] with 99.47% which, though quite close to our results, comes at the cost of more complex and expensive networks.

Table 14 shows inference time and energy consumed per image on both a CPU and GPU processor, along with the storage size of the original and pruned networks. To ensure that the work was applicationally related, we used a test batch of just one image, which is more applicable to real-world testing. Looking at the results on the AID dataset, the storage sizes of both networks are greatly reduced by PulseNetOne, and on a CPU inference is approximately 3× faster, and approximately 10× faster on a GPU. The energy saving for the networks ranges from 1.5× to 13× fewer milli-Joules. Next the results for the MIT67 dataset, where the storage size of AlexNet was reduced in size by nearly 44× while VGG16 was reduced by 12×. The energy saved on the CPU and GPU was 11× and 5.5–11× respectively, while the speed-up in inference time on AlexNet was 3–10× and on VGG16 3–7×.

Table 11. A comparison between state-of-the-art and PulseNetOne results on the SUN397 dataset. The entries in bold show the method with the best classification for the network.

Method	Year	Accuracy
AlexNet fine-tuned on Imagenet [52]	2014	42.61
Otc and HOG [51]	2014	49.60
Hybrid-CNN [52]	2014	53.86
AlexNet fine-tuned on Place205 [52]	2014	54.32
GoogLeNet fine-tuned on Imagenet [53]	2014	55.78
DSP [54]	2014	59.78
GoogLeNet fine-tuned on Place205 [53]	2014	61.51
InterActive [55]	2016	62.97
G-MS2F (ADD) [33]	2017	63.84
G-MS2F (Prod) [33]	2017	64.06
Sparse Representation [57]	2017	71.08
MFAFVNet+Places [58]	2017	72.01
MP [56]	2017	72.60
SDO [29]	2018	73.41
Adi-Red [60]	2018	73.59
SRG [38]	2019	74.06
FOSNet CCG [30]	2019	76.62
FOSNet CCM-CCG [30]	2019	77.28
SOSF+CFA+GAF [50]	2018	78.93
AlexNet-PulseNetOne	2020	**82.11**
VGG16-PulseNetOne	2020	**84.32**

Table 12. Overall accuracies and standard deviations (%) of different CNNs methods along with PulseNetOne on the UC Merced dataset. The entries in bold show the method with the best classification for the network, while the percentage of the original network retained is highlighted in red.

Method	Network Structure	# Parameters	# FLOPs	Accuracy
AlexNet				
scratch	96 − 256 − 384 − 384 − 256 − 4096 − 4096	58367381 (100%)	116715592 (100%)	83.25 ± 0.26
Transferred	-	-	-	94.42 ± 0.18
Fine-tuned	-	-	-	98.19 ± 0.24
PulseNetOne	39 − 59 − 132 − 141 − 106 − 340 − 549	1940812 (3.33%)	3878858 (3.32%)	**99.82 ± 0.11**
VGG16				
scratch	64 − 64 − 128 − 128 − 256 − 256 − 256 − 512− 512 − 512 − 512 − 512 − 512 − 4096 − 4096	134346581 (100%)	268668304 (100%)	75.01 ± 0.32
Transferred	-	-	-	95.24 ± 0.10
Fine-tuned	-	-	-	98.14 ± 0.10
PulseNetOne	10 − 20 − 31 − 29 − 70 − 79 − 83 − 185− 160 − 165 − 172 − 13 − 55 − 272 − 288	1886045 (1.40%)	3768800 (1.40%)	**99.69 ± 0.13**

The results for the NWPU-RESISC45 dataset show the storage sizes of AlexNet and VGG16 were reduced by 44× and 33× respectively, as shown in Table 14. The speed-up in inference time on both networks were between 3× and 10× on the CPU and GPU, respectively. We see from the results on the SCENE15 dataset that the storage sizes of AlexNet and VGG16 were reduced by 77× and 33× respectively, as seen in Table 14. The improvement of the inference timing on both neworks were

3× and 12× for the CPU and GPU, respectively. Table 14 also shows that the CPU energy saving is between 2× and 4×, and between 14.5× and 16× for the GPU.

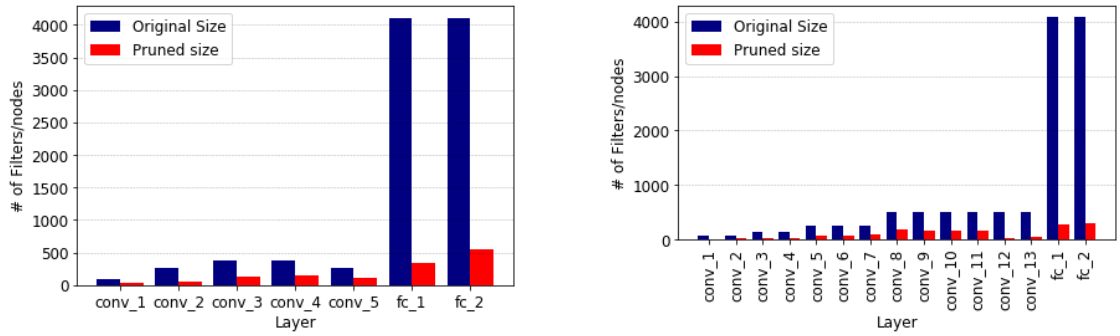

Figure 11. Comparison of layers between original and pruned version of both AlexNet and VGG16 on the UC Merced dataset.

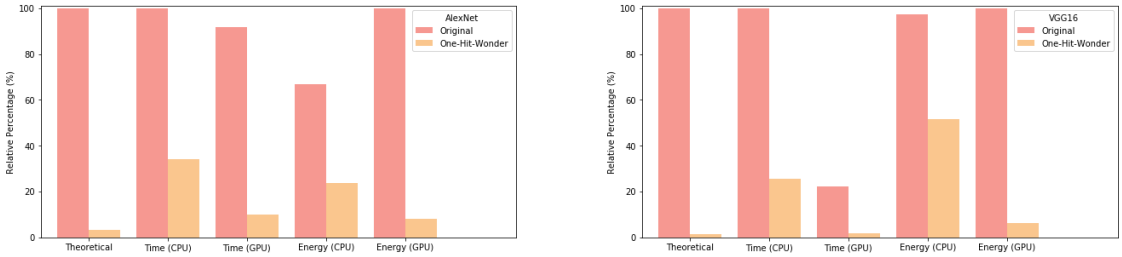

Figure 12. Barcharts for AlexNet and VGG16, illustrating the difference between the theoretical, CPU and GPU efficiency with respect to their computational speed and energy consumed on the UC Merced dataset.

Table 13. A comparison between state-of-the-art and PulseNetOne results on the UCM dataset. The entries in bold show the method with the best classification for the network.

Method	Year	Accuracy
TF-CNN [61]	2016	89.90
Multiview deep learning [62]	2015	93.48
GBRCN [9]	2015	94.53
CNN with OverFeat [63]	2015	95.48
LGF [31]	2016	95.48
Pre-trained AlexNet SPP [32]	2017	95.95
Pre-trained AlexNet SPP-SS [32]	2017	96.67
Discrimative CNNs AlexNet [29]	2018	96.67
Discrimative CNNs GoogLeNet [29]	2018	97.07
Fusion by addition [36]	2017	97.42
Pre-trained AlexNet 1st-FC [64]	2015	98.49
VGG16-CapsNet [35]	2019	98.81
Discrimative CNNs VGG16 [29]	2018	98.93
GCFs+LOFs [48]	2018	99.00
Inception-v3-CapsNet [35]	2019	99.05
Fine-tuned GoogLeNet with SVM [41]	2017	99.47
VGG16-PulseNetOne	2020	**99.69**
AlexNet-PulseNetOne	2020	**99.82**

Next looking at the SUN397 dataset, we see that the storage sizes of AlexNet and VGG16 were reduced 54× and 87× respectively, as shown in Table 14. The improvement in inference timing on both networks were 3.5× and 10× for the CPU and GPU, respectively. As shown in Table 14 the CPU energy saving is 2–3× while the GPU's energy saving is 13.5–17×. Finally on the UCM dataset, it can be seen that the storage sizes of AlexNet and VGG16 were reduced from 222.654 MB to 7.404 MB and 512.492 MB to 7.195 MB, respectively. The speed-up in inference time on both networks were 3.5× and 12× on the CPU and GPU, respectively.

Table 14. Computational results for datasets AID, MIT67, NWPU-RESISC45, Scene15, SUN397 and UC Merced using AlexNet and VGG16. It shows the storage space each network requires, the inference speed per image and the energy consumed per image of both the original and pruned networks using a batch size of a single image.

			Original			PulseNetOne		
Dataset	**CNN**	**Processor**	**Storage (MB)**	**Energy per image (mJ)**	**Time per image (ms)**	**Storage (MB)**	**Energy (mJ)**	**Time per image (ms)**
AID	AlexNet	CPU	222.795	1.47	46.16	5.89	0.51	15.89
-	-	GPU	-	2.68	43.46	-	0.19	4.44
-	VGG16	CPU	512.632	6.59	362.77	26.484	4.89	165.82
-	-	GPU	-	5.88	81.26	-	0.88	10.30
MIT67	AlexNet	CPU	223.373	1.67	46.07	5.105	0.16	17.11
-	-	GPU	-	2.42	42.79	-	0.14	4.10
-	VGG16	CPU	513.21	5.85	364.15	42.388	4.70	109.19
-	-	GPU	-	5.83	81.51	-	1.02	12.28
NWPU-RESISC45	AlexNet	CPU	223.029	1.69	45.78	5.105	0.35	13.24
-	-	GPU	-	2.48	42.37	-	0.15	4.07
-	VGG16	CPU	512.867	5.05	362.75	15.541	5.02	127.78
-	-	GPU	-	5.34	81.49	-	0.55	8.71
Scene15	AlexNet	CPU	222.56	1.66	45.79	2.899	0.45	14.76
-	-	GPU	-	2.24	42.24	-	0.14	3.65
-	VGG16	CPU	512.398	6.54	362.50	15.562	3.84	101.28
-	-	GPU	-	6.05	81.19	-	0.42	7.98
SUN397	AlexNet	CPU	228.53	1.75	46.02	4.218	0.57	16.21
-	-	GPU	-	2.42	43.10	-	0.18	4.52
-	VGG16	CPU	518.368	6.54	363.69	5.961	3.59	96.20
-	-	GPU	-	5.66	81.81	-	0.33	7.41
UC Merced	AlexNet	CPU	222.654	1.70	45.92	7.404	0.60	15.62
-	-	GPU	-	2.55	42.03	-	0.21	4.57
-	VGG16	CPU	512.492	5.79	362.45	7.195	3.06	92.31
-	-	GPU	-	5.95	80.41	-	0.38	6.65

5. Discussion

PulseNetOne removes redundant filters in the convolutional layers (which helps to greatly improve inference timings) and nodes in the fully connected layers (which helps to reduce storage cost). In this sense it can be considered a two-pronged attack on the network architecture, resulting in smaller and more efficient networks with better generalization largely caused (we believe) by a reduction in overfitting. This helps the pruned networks to achieve state-of-the-art results in all the tested remote-sensing benchmark datasets, at a fraction of the computational expense.

All the remote-sensing benchmark datasets used in this research were pruned in a similar fashion: the first and last few convolutional layers contain most redundant filters which are pruned, while the center layers seem to carry more important details for the classification of the datasets (see Figures 1, 3, 5, 7, 9 and 11). In all the datasets the fully connected layers were pruned significantly, confirming a theory in the literature that these layers are over-parameterized (hence some of the newer networks do not include them). We believe that although they are over-parameterized, they can still add value if intelligently pruned.

The experiments had a faster inference time and consumed less energy, due to this, on the GPU compared to the CPU. However, in a real-world environment, it is unlikely that inference would be run on a GPU, rather on a CPU. Therefore, looking at the CPU experiments of both the original and PulseNetOne networks, it can be clearly seen that the pruned networks are significantly more efficient in all manners: storage, speed and energy.

Finally, from Tables 3, 5, 7, 9, 11 and 13 it can be seen that PulseNetOne obtains new state-of-the-art classification accuracy on all the remote-sensing benchmark datasets. Our proposed method consistently outperforms current approaches by between 2% and 4% in most cases.

6. Conclusions

CNNs are state-of-the-art models for image classification, and although much success has been achieved using them in the remote sensing research area, our proposed method shows that the models being used as feature extractors, or as components of other techniques, are highly over-parameterized. We show that by pruning redundant filters and nodes, not only do we achieve better classification accuracy due to a strong regularization on the model, we also create a much more efficient network. PulseNetOne compresses AlexNet and VGG16 on average down to approximately 2% and 4% respectively of their original sizes. Its robustness is demonstrated using six remote- sensing benchmark datasets, on which it greatly compresses the CNNs and achieves state-of-the-art classification accuracy.

Author Contributions: Conceptualization, D.B., M.G. and S.P.; Data curation, D.B.; Investigation, D.B.; Methodology, D.B., M.G. and S.P.; Supervision, S.P.; Validation, D.B.; Writing—original draft, D.B.; Writing—review & editing, D.B. and S.P. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289-P2, and in part by United Technologies Research Center.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Penatti, O.A.; Nogueira, K.; Dos Santos, J.A. Do deep features generalize from everyday objects to remote sensing and aerial scenes domains? In Proceedings of the IEEE conference on computer vision and pattern recognition workshops, Boston, MA, USA, 7–12 June 2015; pp. 44–51.
2. dos Santos, J.A.; Penatti, O.A.B.; da Silva Torres, R. Evaluating the Potential of Texture and Color Descriptors for Remote Sensing Image Retrieval and Classification. In Proceedings of the Fifth International Conference on Computer Vision Theory and Applications, VISAPP (2), Angers, France, 17–21 May 2010; pp. 203–208.
3. Khan, N.Y.; McCane, B.; Wyvill, G. SIFT and SURF performance evaluation against various image deformations on benchmark dataset. In Proceedings of the IEEE 2011 International Conference on Digital Image Computing: Techniques and Applications, Noosa, Australia, 6–8 December 2011; pp. 501–506.

4. Cheng, G.; Zhou, P.; Yao, X.; Yao, C.; Zhang, Y.; Han, J. Object detection in VHR optical remote sensing images via learning rotation-invariant HOG feature. In Proceedings of the IEEE 2016 4th International Workshop on Earth Observation and Remote Sensing Applications (EORSA), Guangzhou, China, 4–6 July 2016; pp. 433–436.
5. Bahmanyar, R.; Cui, S.; Datcu, M. A comparative study of bag-of-words and bag-of-topics models of EO image patches. *IEEE Geosci. Remote Sens. Lett.* **2015**, *12*, 1357–1361. [CrossRef]
6. Huang, L.; Chen, C.; Li, W.; Du, Q. Remote sensing image scene classification using multi-scale completed local binary patterns and fisher vectors. *Remote Sens.* **2016**, *8*, 483. [CrossRef]
7. Zhao, B.; Zhong, Y.; Zhang, L. A spectral–Structural bag-of-features scene classifier for very high spatial resolution remote sensing imagery. *ISPRS J. Photogramm. Remote Sens.* **2016**, *116*, 73–85. [CrossRef]
8. Zhu, Q.; Zhong, Y.; Zhao, B.; Xia, G.S.; Zhang, L. Bag-of-visual-words scene classifier with local and global features for high spatial resolution remote sensing imagery. *IEEE Geosci. Remote Sens. Lett.* **2016**, *13*, 747–751. [CrossRef]
9. Zhang, F.; Du, B.; Zhang, L. Scene classification via a gradient boosting random convolutional network framework. *IEEE Trans. Geosci. Remote Sens.* **2015**, *54*, 1793–1802. [CrossRef]
10. Zhang, L.; Zhang, L.; Du, B. Deep learning for remote sensing data: A technical tutorial on the state of the art. *IEEE Geosci. Remote Sens. Mag.* **2016**, *4*, 22–40. [CrossRef]
11. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, CA, USA, 3–6 December 2012; pp. 1097–1105.
12. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
13. Gong, Y.; Wang, L.; Guo, R.; Lazebnik, S. Multi-scale orderless pooling of deep convolutional activation features. In *European Conference on Computer Vision*; Springer: Berlin, Germany, 2014, pp. 392–407.
14. Zuo, Z.; Wang, G.; Shuai, B.; Zhao, L.; Yang, Q.; Jiang, X. Learning discriminative and shareable features for scene classification. In *European Conference on Computer Vision*; Springer: Berlin, Germany, 2014; pp. 552–568.
15. Wu, R.; Wang, B.; Wang, W.; Yu, Y. Harvesting discriminative meta objects with deep CNN features for scene classification. In Proceedings of the IEEE International Conference on Computer Vision, Beijing, China, 21 October 2015; pp. 1287–1295.
16. Herranz, L.; Jiang, S.; Li, X. Scene recognition with CNNs: Objects, scales and dataset bias. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 571–579.
17. Cheriyadat, A.M. Unsupervised feature learning for aerial scene classification. *IEEE Trans. Geosci. Remote Sens.* **2013**, *52*, 439–451. [CrossRef]
18. Dalal, N.; Triggs, B. Histograms of oriented gradients for human detection. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 20–25 June 2005.
19. Lowe, D.G. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* **2004**, *60*, 91–110. [CrossRef]
20. Csurka, G.; Dance, C.; Fan, L.; Willamowski, J.; Bray, C. Visual categorization with bags of keypoints. In Proceedings of the Workshop on Statistical Learning in Computer Vision, ECCV, Prague, Czech Republic, 11–14 May 2004; Volume 1, pp. 1–2.
21. Bian, X.; Chen, C.; Tian, L.; Du, Q. Fusing local and global features for high-resolution scene classification. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2017**, *10*, 2889–2901. [CrossRef]
22. Lu, X.; Li, X.; Mou, L. Semi-supervised multitask learning for scene recognition. *IEEE Trans. Cybern.* **2014**, *45*, 1967–1976. [PubMed]
23. Chen, C.; Zhang, B.; Su, H.; Li, W.; Wang, L. Land-use scene classification using multi-scale completed local binary patterns. *Signal Image Video Process.* **2016**, *10*, 745–752. [CrossRef]
24. Lazebnik, S.; Schmid, C.; Ponce, J. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), New York, NY, USA, 17–22 June 2006; Volume 2, pp. 2169–2178.

25. Yang, Y.; Newsam, S. Comparing SIFT descriptors and Gabor texture features for classification of remote sensed imagery. In Proceedings of the 2008 15th IEEE International Conference on Image Processing, San Diego, CA, USA, 12–15 October 2008; pp. 1852–1855.
26. Liu, Q.; Hang, R.; Song, H.; Li, Z. Learning multiscale deep features for high-resolution satellite image scene classification. *IEEE Trans. Geosci. Remote Sens.* **2017**, *56*, 117–126. [CrossRef]
27. Sheng, G.; Yang, W.; Xu, T.; Sun, H. High-resolution satellite scene classification using a sparse coding based multiple feature combination. *Int. J. Remote Sens.* **2012**, *33*, 2395–2412. [CrossRef]
28. Zhang, F.; Du, B.; Zhang, L. Saliency-guided unsupervised feature learning for scene classification. *IEEE Trans. Geosci. Remote Sens.* **2014**, *53*, 2175–2184. [CrossRef]
29. Cheng, G.; Yang, C.; Yao, X.; Guo, L.; Han, J. When deep learning meets metric learning: Remote sensing image scene classification via learning discriminative CNNs. *IEEE Trans. Geosci. Remote Sens.* **2018**, *56*, 2811–2821. [CrossRef]
30. Seong, H.; Hyun, J.; Kim, E. FOSNet: An End-to-End Trainable Deep Neural Network for Scene Recognition. *arXiv* **2019**, arXiv:1907.07570.
31. Zou, J.; Li, W.; Chen, C.; Du, Q. Scene classification using local and global features with collaborative representation fusion. *Inf. Sci.* **2016**, *348*, 209–226. [CrossRef]
32. Han, X.; Zhong, Y.; Cao, L.; Zhang, L. Pre-trained alexnet architecture with pyramid pooling and supervision for high spatial resolution remote sensing image scene classification. *Remote Sens.* **2017**, *9*, 848. [CrossRef]
33. Tang, P.; Wang, H.; Kwong, S. G-MS2F: GoogLeNet based multi-stage feature fusion of deep CNN for scene recognition. *Neurocomputing* **2017**, *225*, 188–197. [CrossRef]
34. Liu, B.D.; Xie, W.Y.; Meng, J.; Li, Y.; Wang, Y. Hybrid collaborative representation for remote-sensing image scene classification. *Remote Sens.* **2018**, *10*, 1934. [CrossRef]
35. Zhang, W.; Tang, P.; Zhao, L. Remote Sensing Image Scene Classification Using CNN-CapsNet. *Remote Sens.* **2019**, *11*, 494. [CrossRef]
36. Chaib, S.; Liu, H.; Gu, Y.; Yao, H. Deep feature fusion for VHR remote sensing scene classification. *IEEE Trans. Geosci. Remote Sens.* **2017**, *55*, 4775–4784. [CrossRef]
37. Sun, Q.S.; Zeng, S.G.; Liu, Y.; Heng, P.A.; Xia, D.S. A new method of feature fusion and its application in image recognition. *Pattern Recognit.* **2005**, *38*, 2437–2448. [CrossRef]
38. Zeng, H.; Chen, G. Scene Recognition with Comprehensive Regions Graph Modeling. In *International Conference on Image and Graphics*; Springer: Berlin, Germany, 2019, pp. 630–641.
39. Kipf, T.N.; Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv* **2016**, arXiv:1609.02907.
40. Han, D.; Liu, Q.; Fan, W. A new image classification method using CNN transfer learning and web data augmentation. *Expert Syst. Appl.* **2018**, *95*, 43–56. [CrossRef]
41. Nogueira, K.; Penatti, O.A.; dos Santos, J.A. Towards better exploiting convolutional neural networks for remote sensing scene classification. *Pattern Recognit.* **2017**, *61*, 539–556. [CrossRef]
42. Browne, B.; Giering, G.; Prestwich, P. Pulse-Net: Dynamic Compression of Convolutional Neural Networks. In Proceedings of the 2019 IEEE 5th World Forum on Internet of Things (WF-IoT), Limerick, Ireland, 15–18 April 2019; pp. 346–351.
43. Yang, Y.; Newsam, S. Bag-of-visual-words and spatial extensions for land-use classification. In Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems, Seattle, WA, USA, 4–6 November 2010; pp. 270–279.
44. Cheng, G.; Han, J.; Lu, X. Remote sensing image scene classification: Benchmark and state of the art. *Proc. IEEE* **2017**, *105*, 1865–1883. [CrossRef]
45. Xiao, J.; Hays, J.; Ehinger, K.A.; Oliva, A.; Torralba, A. Sun database: Large-scale scene recognition from abbey to zoo. In Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Francisco, CA, USA, 13–18 June 2010; pp. 3485–3492.
46. Quattoni, A.; Torralba, A. Recognizing indoor scenes. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 413–420.
47. Xia, G.S.; Hu, J.; Hu, F.; Shi, B.; Bai, X.; Zhong, Y.; Zhang, L.; Lu, X. AID: A benchmark data set for performance evaluation of aerial scene classification. *IEEE Trans. Geosci. Remote Sens.* **2017**, *55*, 3965–3981. [CrossRef]

48. Chen, G.; Zhang, X.; Tan, X.; Cheng, Y.; Dai, F.; Zhu, K.; Gong, Y.; Wang, Q. Training small networks for scene classification of remote sensing images via knowledge distillation. *Remote Sens.* **2018**, *10*, 719. [CrossRef]
49. Yu, Y.; Liu, F. A two-stream deep fusion framework for high-resolution aerial scene classification. *Comput. Intell. Neurosci.* **2018**, *2018*. [CrossRef] [PubMed]
50. Sun, N.; Li, W.; Liu, J.; Han, G.; Wu, C. Fusing Object Semantics and Deep Appearance Features for Scene Recognition. *IEEE Trans. Circuits Syst. Video Technol.* **2018**, *29*, 1715–1728. [CrossRef]
51. Margolin, R.; Zelnik-Manor, L.; Tal, A. Otc: A novel local descriptor for scene classification. In *European Conference on Computer Vision*; Springer: Berlin, Germany, 2014; pp. 377–391.
52. Zhou, B.; Lapedriza, A.; Xiao, J.; Torralba, A.; Oliva, A. Learning deep features for scene recognition using places database. In Proceedings of the Advances in neural information processing systems, Montreal, QC, Canada, 8–13 December 2014; pp. 487–495.
53. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, CA, USA, 7–12 June 2015; pp. 1–9.
54. Gao, B.B.; Wei, X.S.; Wu, J.; Lin, W. Deep spatial pyramid: The devil is once again in the details. *arXiv* **2015**, arXiv:1504.05277.
55. Xie, L.; Zheng, L.; Wang, J.; Yuille, A.L.; Tian, Q. Interactive: Inter-layer activeness propagation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 270–279.
56. Song, X.; Jiang, S.; Herranz, L. Multi-scale multi-feature context modeling for scene recognition in the semantic manifold. *IEEE Trans. Image Process.* **2017**, *26*, 2721–2735. [CrossRef] [PubMed]
57. Nascimento, G.; Laranjeira, C.; Braz, V.; Lacerda, A.; Nascimento, E.R. A robust indoor scene recognition method based on sparse representation. In *Iberoamerican Congress on Pattern Recognition*; Springer: Berlin, Germany, 2017, pp. 408–415.
58. Li, Y.; Dixit, M.; Vasconcelos, N. Deep scene image classification with the MFAFVNet. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 5746–5754.
59. Liu, Y.; Huang, C. Scene classification via triplet networks. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2017**, *11*, 220–237. [CrossRef]
60. Zhao, Z.; Larson, M. From Volcano to Toyshop: Adaptive Discriminative Region Discovery for Scene Recognition. *arXiv* **2018**, arXiv:1807.08624.
61. Zhong, Y.; Fei, F.; Zhang, L. Large patch convolutional neural networks for the scene classification of high spatial resolution imagery. *J. Appl. Remote Sens.* **2016**, *10*, 025006. [CrossRef]
62. Luus, F.P.; Salmon, B.P.; Van den Bergh, F.; Maharaj, B.T.J. Multiview deep learning for land-use classification. *IEEE Geosci. Remote Sens. Lett.* **2015**, *12*, 2448–2452. [CrossRef]
63. Marmanis, D.; Datcu, M.; Esch, T.; Stilla, U. Deep learning earth observation classification using ImageNet pretrained networks. *IEEE Geosci. Remote Sens. Lett.* **2015**, *13*, 105–109. [CrossRef]
64. Hu, F.; Xia, G.S.; Hu, J.; Zhang, L. Transferring deep convolutional neural networks for the scene classification of high-resolution remote sensing imagery. *Remote Sens.* **2015**, *7*, 14680–14707. [CrossRef]